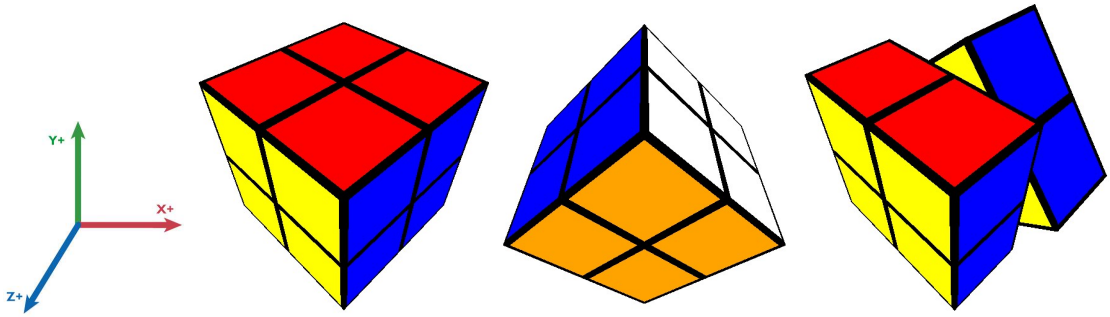
Rubiks (rubiks.*) Useu: `~/assig/grau-g/Viewer/GLarenaSL`

Escriu **VS+GS+FS** per tal de dibuixar un cub de Rubik simplificat:



El **VS** farà les tasques imprescindibles.

El **GS** serà l'encarregat de dibuixar els cubs (un cub per cada execució del GS). El GS sabrà quin dels 8 cubs ha d'emetre pel valor de gl_PrimitiveIDIn. Si gl_PrimitiveIDIn >= 8, el GS no emetrà cap primitiva.

Els cubs tindran longitud d'aresta 2, i estaran centrats en els següents punts (en *object space*):

`(-1,-1,-1), (1,-1,-1), (-1,1,-1), (1,1,-1), (-1,-1, 1), (1,-1, 1), (-1,1,1), (1,1,1)`

El color de les cares del cub serà, segons l'orientació de la seva normal (tot i que no usareu il·luminació):

bottom (y negativa):	orange RGB (1, 0.6, 0);	top (y positiva): red
left (x negativa):	green	right (x positiva): blue
back (z negativa):	white	front (z positiva): yellow

Abans de passar els vèrtexs a *clip space*, el GS aplicarà una rotació de θ=*time* radians al voltant de l'eix Z, però només si gl_PrimitiveIDIn < 4 (és a dir, per 4 dels 8 cubs). Recordeu que la matriu de rotació al voltant de l'eix Z és:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

El **FS** assignarà al fragment el color que li arribi del GS, tret d'un petit marge al voltant de cada cara, que serà de color negre. Per aquest marge, podeu fer que el GS generi coordenades de textura (s,t) per cada vèrtex, entre 0 i 1, i que el FS utilitzi el color de la cara si s, t estan dins [0.05, 0.95]; altrament, el color serà negre. Puntuació orientativa:

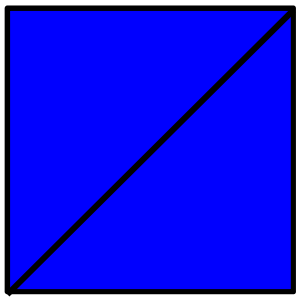
- Cubs amb la mida i posició que es demana: 5 punts
- Cares amb els colors que es demanen: 2 punts
- Marge negre a les cares: 2 punts
- Rotació: 1 punt

Identificadors obligatoris:

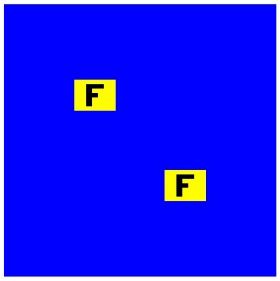
`rubiks.vert, rubiks.geom, rubiks.frag` (segur que has escrit **rubiks** correctament?)
Tots els uniforms de l'enunciat.

Labeler (labeler.*) Useu: `~/assig/grau-g/Viewer/GLarenaSL`

Escriu **VS+GS+FS** per tal de dibuixar un petit rectangle amb la lletra "F" al centre de cada triangle:



Triangles de l'objecte plane.obj



Resultat esperat

El **VS** farà les tasques imprescindibles.

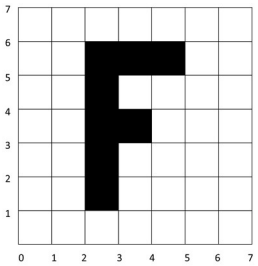
El **GS** farà les tasques per defecte (dibuixar el triangle original) i a més a més serà l'encarregat de dibuixar el rectangle amb la lletra F. Aquest rectangle estarà centrat al centre del triangle.

Usarem aquests dos uniforms:

```
uniform float size = 0.07;
uniform float depth = -0.01;
```

La mida del rectangle serà de 2*size, on size està en NDC (Normalized Device Coordinates, amb valors en [-1, 1] per punts dins la piràmide de visió de la càmera). També introduirem un petit offset a la z per tal d'evitar que el triangle oculti part del rectangle. Per tant, si C és el centre del triangle en NDC, els vèrtexs del rectangle que cal emetre al GS tindran coordenades de la forma (C.x ± size, C.y ± size, C.z + depth, 1.0). Observa que hem afegit una w=1.0 perquè la sortida del GS cal que sigui en clip space.

El **FS** assignarà al fragment el color que li arribi del GS, tret de quan el fragment pertany al rectangle afegit. En aquest cas, cal texturar el fragment de forma procedural, a partir de les coordenades de textura que rebrà del GS. Assumint que teniu coordenades de textura en [0,7], aquí teniu els fragments que haran de ser negres (la resta seran de color groc):



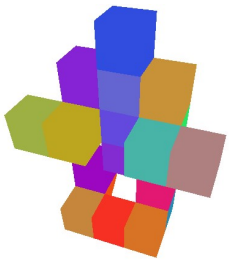
Identificadors obligatoris:

`labeler.vert, labeler.geom, labeler.frag` (segur que has escrit **labeler** correctament?)
Tots els uniforms de l'enunciat.

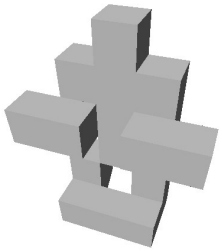
Minecraft (minecraft.*)

Useu: `~/assig/grau-g/Viewer/GLarenaSL`

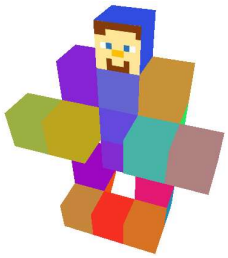
Escriu **VS+GS+FS** per mostrar una representació del model (**man.obj**) amb capses:



mode=0



mode=1



mode=2

face.jpg

El VS farà les tasques imprescindibles, però aplicant al model un **escalat uniforme de 3**. La posició del vèrtex s'escriurà en o*bject space*. El color de sortida serà el color original del vèrtex, sense il·luminació.

El GS només processarà un de cada N triangles, d'acord amb un **uniform int N = 300.** Concretament, ignorarà tots els triangles excepte els triangles amb gl_PrimitiveIDIn = 0, N, 2N, 3N... Pels triangles no ignorats, el GS generarà un cub. Sigui C la mitjana dels vèrtexs del triangle (en *object space*). El cub tindrà mida 1 i estarà centrat al punt amb coordenades enteres més proper al punt C. Recordeu que els vèrtexs de sortida del GS han d'estar en *clip space*. El color de la cara d'un cub que calcularà el GS dependrà d'un **uniform int mode = 0**.

[4 punts] **Si mode = 0**, el color de cada cub serà la mitjana dels colors dels vèrtexs (sense il·luminació).

[4 punts] **Si mode = 1**, el color serà el gris (N.z, N.z, N.z), on N és la normal de la cara del cub, en *eye space,* calculada al GS.

[2 punts] **Si mode = 2**, llavors el color es calcularà com al mode 0 excepte per les cares amb una normal apuntant cap a Z+ que provenen de cubs creats per triangles amb **C.y >=4.5.** En aquest cas, el GS generarà coordenades de textura per tal que el FS pugui texturar la cara (imatge de dalt a la dreta).

El FS usarà el color del GS o el color d'una textura **uniform sampler2D face,** segons el mode i la cara.

Identificadors obligatoris:
`minecraft.vert, minecraft.geom, minecraft.frag`
Tots els uniform's de l'enunciat.